head 1.2; access; symbols; locks; strict; comment @# @; 1.2 date 97.02.15.02.19.41; author peercy; state Exp; branches; next 1.1; 1.1 date 97.01.24.01.09.28; author airey; state Exp; branches; next ; desc @@ 1.2 log @cleanup pages and restructure them so they are spec-centric @ text @

TINY FLOATING POINT

Two basic disadvantages of fixed-point arithmetic are: (1) The range of numbers that can be represented is small e.g. with a L bits signed two's complement fixed format, the smallest number is -1 and the largest is 1-s^-L. (2) The relative error, which can be thought of as percentage error, increases as the magnitude of the number is decreased. A floating point format in general leads to increased dynamic range and constant relative error.

This section describes a tiny IEEE like floating point format; tiny because the total number of bits is less than IEEE32. The errors, the implication of trading mantissa and exponent bits, and selection of a bias is discussed in the following sections. Other non-fixed point format such as the compressed Z or logarithymic may be considered in the future, tiny IEEE is chosen to be the prime candidate because of it's well understood behavior and hardware implementation.

The discussion to follow uses L=16 as examples because 16 bits is most likely the basic unit of transfer among memory and other subsystems.

Relative Error

The relative error is the ratio of the absolute error i.e. the difference between a value x and the corresponding quantized value, to the value x.

Given L bits in a fixed-point format, the absolute error is (1/2)2^(-L) / x. In other words, the relative error of a fixed point representation is larger when the represented value is less than 1 than when it is greater than 1.

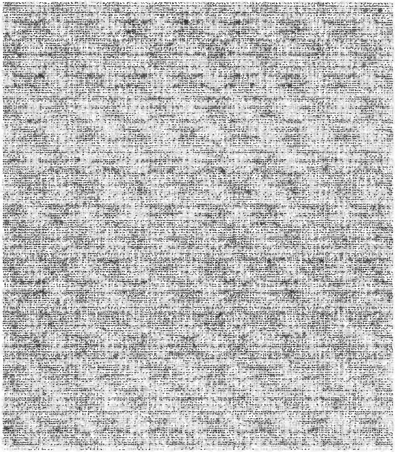
In contrast, floating point format offers constant relative error (1/2*2^(-L)) where L is the number of

TINY FLOATING POINT

Two basic disadvantages of fixed-point arithmetic are: (1) The range of numbers that can be represented is small e.g. with a L bits signed two's complement fixed format, the smallest number is -1 and the largest is 1-s^-L. (2) The relative error, which can be thought of as percentage error, increases as the magnitude of the number is decreased. A floating point format in general leads to increased dynamic range and constant relative error.

This section describes a tiny IEEE like floating point format; tiny because the total number of bits is less than IEEE32. The errors, the implication of trading mantissa and exponent bits, and selection of a bias is discussed in the following sections. Other non-fixed point format such as the compressed Z or logarithymic may be considered in the future, tiny IEEE is chosen to be the prime candidate because of it's well understood behavior and hardware implementation.

The discussion to follow uses L=16 as examples because 16 bits is most likely the basic unit of transfer among memory and other subsystems.



Relative Error

The relative error is the ratio of the absolute error i.e. the difference between a value x and the corresponding quantized value, to the value x.

Given L bits in a fixed-point format, the absolute error is (1/2)2^(-L) / x. In other words, the relative error of a fixed point representation is larger when the represented value is less than 1 than when it is greater than 1.

In contrast, floating point format offers constant relative error (1/2*2^(-L)) where L is the number of mantissa bits. Therefore, the absolute error of floating point format is smaller when the represented value is less than 1 than when it is greater than 1; a desirable property since we want precision for color values within the displayable range i.e.(0,1) and are willing to trade off that precision in extended

HIGHLY CONFIDENTIAL

range when the value is greater than 1.

The absolute error of **s10e5**is tabulated in Table1 for each of the 32 exponent value. The error is defined as the width of quantization: the smallest number variation that can be represented at the least significant bit of mantissa. It's interesting to observe that for number close to zero, the absoulte error is 2^25 which is way better than a 16 bit fixed number.

Table 1Absolute error of s10e5 in each exponent range.

Absolute error in each range				
	Bias=15	Bias=15	Bias=7	Bias=7
Exp	Range	Absolute error	Range	Absolute error
0	0.000031-0.000061	2^-25	0.007812-0.015617	2^-17
1	0.000061-0.000122	2^-24	0.015625-0.031235	2^-16
2	0.000122-0.000244	2^-23	0.031250-0.062469	2^-15
3	0.000244-0.000488	2^-22	0.062500-0.124939	2^-14
4	0.000488-0.000976	2^-21	0.125000-0.249878	2^-13
5	0.000977-0.001952	2^-20	0.250000-0.499756	2^-12
6	0.001953-0.003904	2^-19	0.500000-0.999512	2^-11
7	0.003906-0.007809	2^-18	1.000000-1.999023	2^-10
8	0.007812-0.015617	2^-17	2.000000-3.998047	2^-9
9	0.015625-0.031235	2^-16	4.000000-7.996094	2^-8
10	0.031250-0.062469	2^-15	8.000000-15.992187	2^-7
11	0.062500-0.124939	2^-14	16.000000-31.984374	2^-6
12	0.125000-0.249878	2^-13	32.000000-63.968749	2^-5
13	0.250000-0.499756	2^-12	64.000000-127.937498	2^-4
14	0.500000-0.999512	2^-11	128.000000-255.874995	2^-3
15	1.000000-1.999023	2^-10	256.000000-511.749990	2^-2
16	2.000000-3.998047	2^-9	512.000000-1023.499981	2^-1
17	4.000000-7.996094	2^-8	1024.000000-2046.999962	2^0
18	8.000000-15.992187	2^-7	2048.000000-4093.999923	2^1
19	16.000000-31.984374	2^-6	4096.000000-8187.999846	2^2

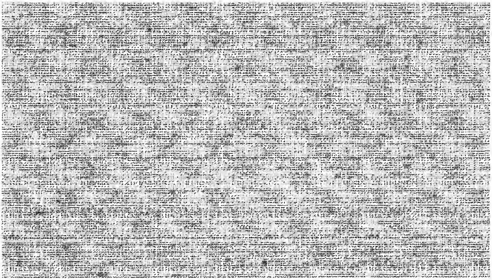
S0134898

20	32.000000-63.968749	2^-5	8192.000000-16375.999693	2^3
21	64.000000-127.937498	2^-4	16384.000000-32751.999386	2^4
22	128.000000-255.874995	2^-3	32768.000000-65503.998771	2^5
23	256.000000-511.749990	2^-2	65536.000000-131007.997542	2^6
24	512.000000-1023.499981	2^-1	131072.000000-262015.995085	2^7
25	1024.000000-2046.999962	2^0	262144.000000-524031.990170	2^8
26	2048.000000-4093.999923	2^1	524288.000000-1048063.980339	2^9
27	4096.000000-8187.999846	2^2	1048576.000000-2096127.960678	2^10
28	8192.000000-16375.999693	2^3	2097152.000000-4192255.921357	2^11
29	16384.000000-32751.999386	2^4	4194304.000000-8384511.842714	2^12
30	32768.000000-65503.998771	2^5	8388608.000000-16769023.685427	2^13
31	65536.000000-131007.997542	2^6	16777216.000000-33538047.370854	2^14

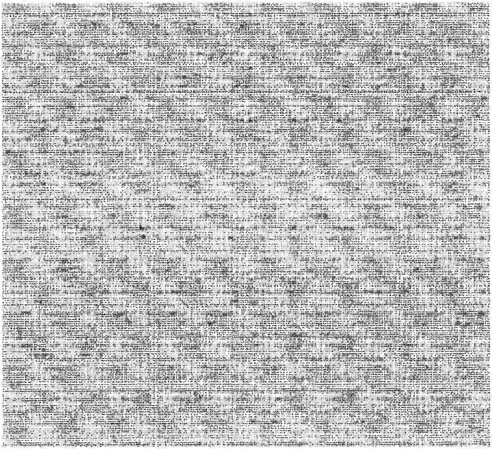
Trade off between exponent and mantissa

Given a fixed number of bits, the partitioning of these bits into either the exponent or mantissa fields becomes an exercise of trading off range with relative error. In general, increasing number of exponent bits and decreasing number of mantissa bits increases range at the expense of increased relative error. In Plot1, the x axis is the 16 bit number interpreted as an unsigned integer and the y -axis is the 16 bit number as floating point. Observe that when a bit is moved from the mantissa field to the exponent field i.e. s10e5 to s9e6 or s11e4 to s10e5, the range is extended on both sides: the largest representable number is s^16 times larger and the smallest representable number is 2^-16 times smaller.

Plot 1



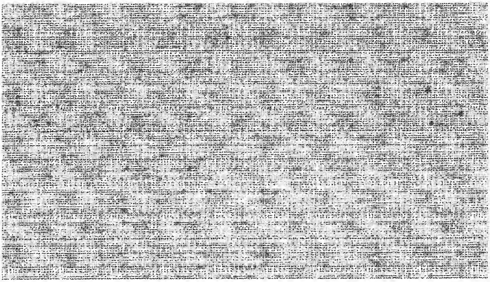
S0134899



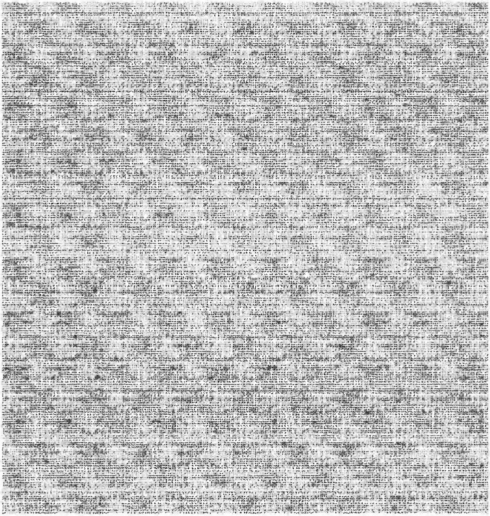
Where do these numbers in the extended range come from ? Histogram 1 gives a picture on how the bits are re-distributed when mantissa bits are trade off with exponent bits. In Histogram 1, the bins are logarithmic in size i.e. bin 0 (0,1), bin 1 (1,2), bin 2 (2, 4) etc. Note that for all three formats in the histogram, the sizes of bin 0 remain the same. As we go from s10e5 to s9e6, the number of numbers between (0,1) remains the same, and some of the numbers which were above 1.0 is re-distributed to the new extended range.

Also note that for s10e5, about half the numbers are within (0,+/-1) range.

Histogram 1



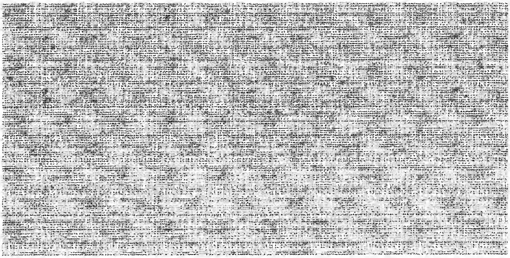
S0134900

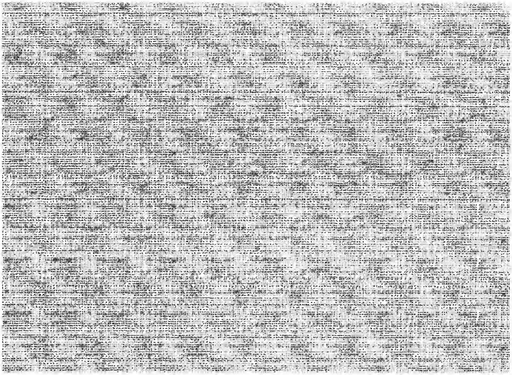


Choosing a bias

Changing the bias can also has an effect on range, but not on relative error. As shown in plot 2, changing the bias from 15 to 14 has double both the largest and the smallest representable number.

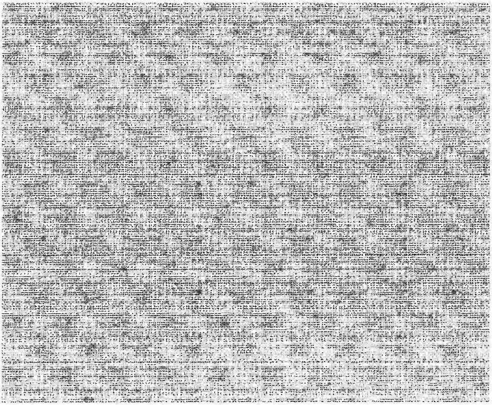
Plot 2

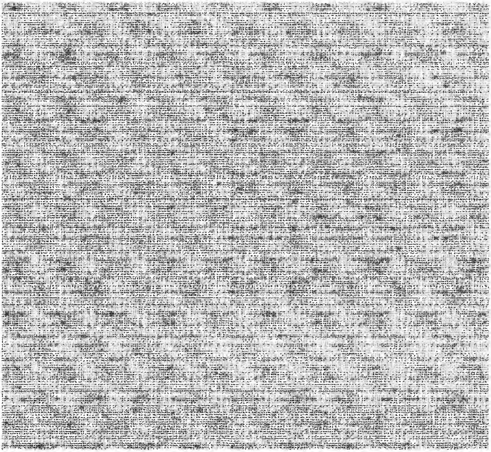




Histogram 2 shows how the numbers are re-distributed when bias is changed. As the bias is changed from 17 to 12, the number of values in bin zero decreases as the range increases while the sizes of all other bins remain constant. Effectively, the range is extended at the expense of having less values between (0.0, 1.0).

Histogram2





S10E5

12 bit fixed versus S10E5

When using S10E5 as the canonical intermediate format in the rchip, one desirable property is to preserve 12 bits fixed accuracy when we convert from 12 bit fixed to S10E5 back to 12 bit fixed.

From Table 1, the absolute error in the range (-.499875, .499875) has more than or equal to 12 bits of absolute error. So we can linearly map (0, 4095) 12 bit fixed to (-.499875, .499875) and back to (0, 4095) without losing any bits. A *simple program* using arith.c verifies that this is true.

S0134903